mdn web docs

Proxy servers and tunneling

When navigating through different networks of the Internet, proxy servers and HTTP tunnels are facilitating access to content on the World Wide Web. A proxy can be on the user's local computer, or anywhere between the user's computer and a destination server on the Internet. This page outlines some basics about proxies and introduces a few configuration options.

There are two types of proxies: **forward proxies** (or tunnel, or gateway) and **reverse proxies** (used to control and protect access to a server for load-balancing, authentication, decryption or caching).

Forward proxies

A forward proxy, or gateway, or just "proxy" provides proxy services to a client or a group of clients. There are likely hundreds of thousands of open forward proxies on the Internet. They store and forward Internet services (like the DNS, or web pages) to reduce and control the bandwidth used by the group.

Forward proxies can also be anonymous and allow users to hide their IP address while browsing the Web or using other Internet services. For example, Tor routes internet traffic through multiple proxies for anonymity.

Reverse proxies

As the name implies, a reverse proxy does the opposite of what a forward proxy does: A forward proxy acts on behalf of clients (or requesting hosts). Forward proxies can hide the identities of clients whereas reverse proxies can hide the identities of servers. Reverse proxies have several use cases, a few are:

- Load balancing: distribute the load to several web servers,
- Cache static content: offload the web servers by caching static content like pictures,
- Compression: compress and optimize content to speed up load time.

Forwarding client information through proxies

Proxies can make requests appear as if they originated from the proxy's IP address. This can be useful if a proxy is used to provide client anonymity, but in other cases information from the original request is lost. The IP address of the original client is often used for debugging, statistics, or generating location-dependent content. A common way to disclose this information is by using the following HTTP headers:

The standardized header:

`Forwarded`

Contains information from the client-facing side of proxy servers that is altered or lost when a proxy is involved in the path of the request.

Or the de-facto standard versions:

`X-Forwarded-For`

Identifies the originating IP addresses of a client connecting to a web server through an HTTP proxy or a load balancer.

`X-Forwarded-Host`

Identifies the original host requested that a client used to connect to your proxy or load balancer.

`X-Forwarded-Proto`

identifies the protocol (HTTP or HTTPS) that a client used to connect to your proxy or load balancer.

To provide information about the proxy itself (not about the client connecting to it), the `Via` header can be used.

`Via`

Added by proxies, both forward and reverse proxies, and can appear in the request headers and the response headers.

HTTP tunneling

Tunneling transmits private network data and protocol information through public network by encapsulating the data. HTTP tunneling is using a protocol of higher level (HTTP) to transport a lower level protocol (TCP).

The HTTP protocol specifies a request method called `CONNECT`. It starts two-way communications with the requested resource and can be used to open a tunnel. This is how a client behind an HTTP proxy can access websites using TLS (i.e., HTTPS, port 443). Note, however, that not all proxy servers suppo method or limit it to port 443 only.

See also the HTTP tunnel article on Wikipedia .

Proxy Auto-Configuration (PAC)

A Proxy Auto-Configuration (PAC) file is a JavaScript function that determines whether web browser requests (HTTP, HTTPS, and FTP) go directly to the destination or are forwarded to a web proxy server. The JavaScript function contained in the PAC file defines the function:

The auto-config file should be saved to a file with a `.pac` filename extension: `proxy.pac`.

And the MIME type set to `application/x-ns-proxy-autoconfig`.

The file consists of a function called `FindProxyForURL`. The example below will work in an environment where the internal DNS server is set up so that it can only resolve internal host names, and the goal is to use a proxy only for hosts that aren't resolvable:

JS

```
function FindProxyForURL(url, host) {
  if (isResolvable(host)) {
    return "DIRECT";
  }
  return "PROXY proxy.mydomain.com:8080";
}
```

See also

- Proxy Auto-Configuration (PAC) file
- `CONNECT` method
- Proxy server